Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

Lab Work
“Lab-3”

[Code No: COMP-342]

Submitted by:

Bibhushan Saakha [41]

Submitted to:

Mr Dhiraj Shrestha
Department of Computer Science and Engineering

Submission Date:
2024-06-02

Questions:

1. Write a Program to implement mid- point Circle Drawing Algorithm
2. Write a Program to implement mid- point Ellipse Drawing Algorithm

(For doing these Transformations consider any 2D shapes (Line, Triangle, Rectangle etc), and use Homogeneous coordinate Systems)

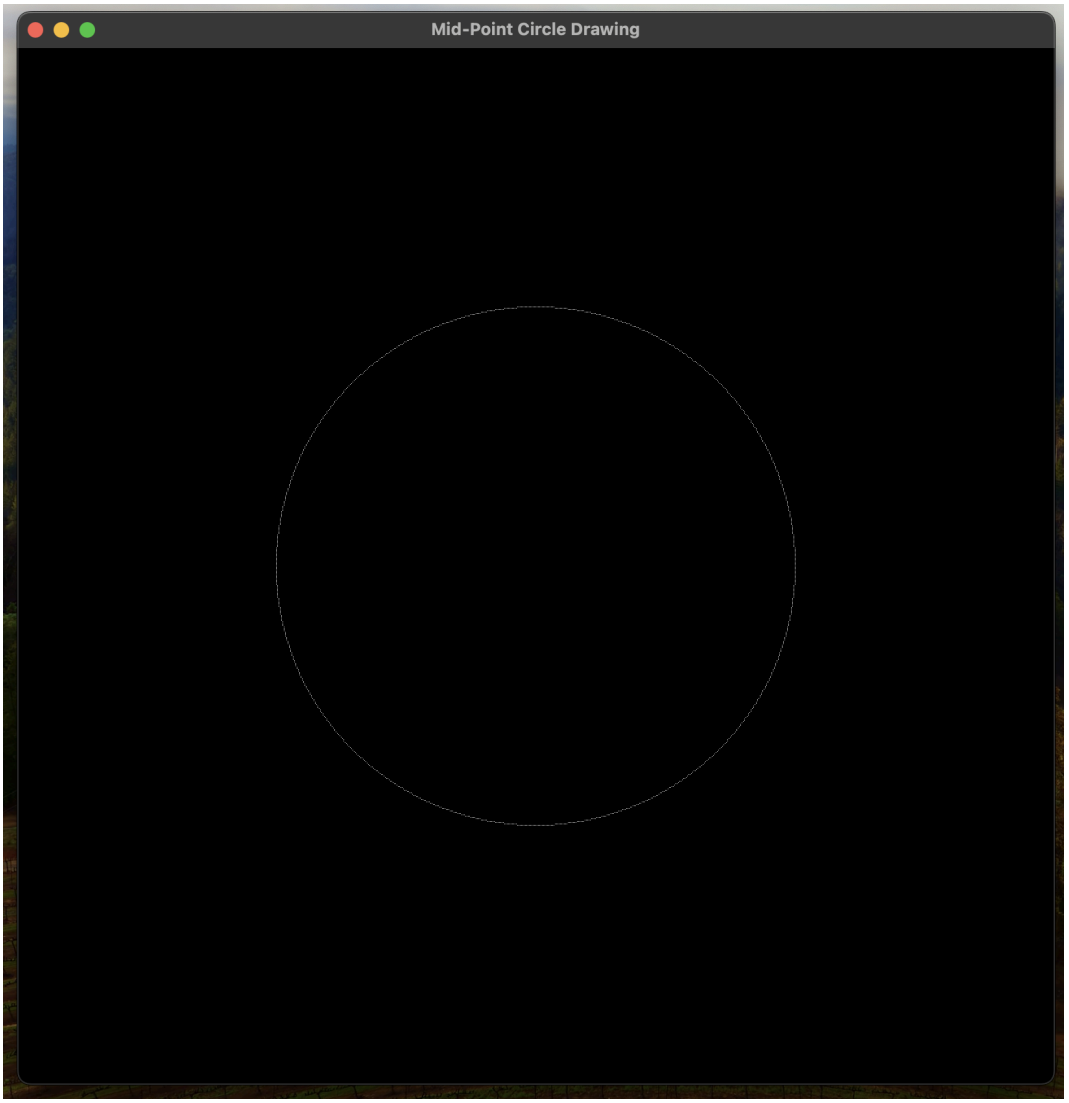
Midpoint Circle Drawing Algorithm Program:

The Mid-Point Circle Drawing Algorithm is a rasterization technique used to compute the pixel coordinates required to draw a circle. This algorithm efficiently calculates the perimeter points of a circle within the first octant and then exploits the circle's symmetry to mirror these points across the remaining seven octants, ensuring a complete and accurate representation of the circle.

```python
import glfw
from OpenGL.GL import *
from OpenGL.GLU import *

win_width, win_height = 800, 800

def MidPointCircle(x_center, y_center, r):
    points = []
    x = 0
    y = r
    P = 1 - r

    while x <= y:
        points.append((x_center + x, y_center + y))
        points.append((x_center - x, y_center + y))
        points.append((x_center + x, y_center - y))
        points.append((x_center - x, y_center - y))
        points.append((x_center + y, y_center + x))
        points.append((x_center - y, y_center + x))
        points.append((x_center + y, y_center - x))
        points.append((x_center - y, y_center - x))

        if P < 0:
            P = P + 2 * (x + 1) + 1
        else:
            P = P + 2 * (x + 1) + 1 - 2 * (y - 1)
            y -= 1

        x += 1

    return points

def draw_circle(x_center, y_center, radius):
    points = MidPointCircle(x_center, y_center, radius)
    glBegin(GL_POINTS)
    glColor3f(1.0, 1.0, 1.0)  # Set color to white
    for x, y in points:
        glVertex2f(x, y)
    glEnd()


```

```python
def main():
    # Initialize glfw
    if not glfw.init():
        return

    # Create a windowed mode window and its OpenGL context
    window = glfw.create_window(win_width, win_height, "Mid-Point Circle Drawing", None,
None)if not window:
        glfw.terminate()
        return

    # Make the window's context current
    glfw.make_context_current(window)

    # Set viewport and projection
    glViewport(0, 0, win_width, win_height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(0, win_width, 0, win_height)

    # Loop until the user closes the window
    while not glfw.window_should_close(window):
        # Render here, e.g. using pyOpenGL
        glClear(GL_COLOR_BUFFER_BIT)
        draw_circle(400, 400, 200)

        # Swap front and back buffers
        glfw.swap_buffers(window)

        # Poll for and process events
        glfw.poll_events()

    glfw.terminate()

if __name__ == "__main__":
    main()

```

Output:



This Python code uses GLFW and OpenGL to draw a circle with the Midpoint Circle Drawing Algorithm. The MidPointCircle function calculates the perimeter points of a circle centered at (x_center, y_center) with radius r. The draw_circle function renders the circle using OpenGL.

In the main function, a GLFW window is initialized, and a circle is drawn at (400, 400) with a radius of 200. The window refreshes until closed by the user.

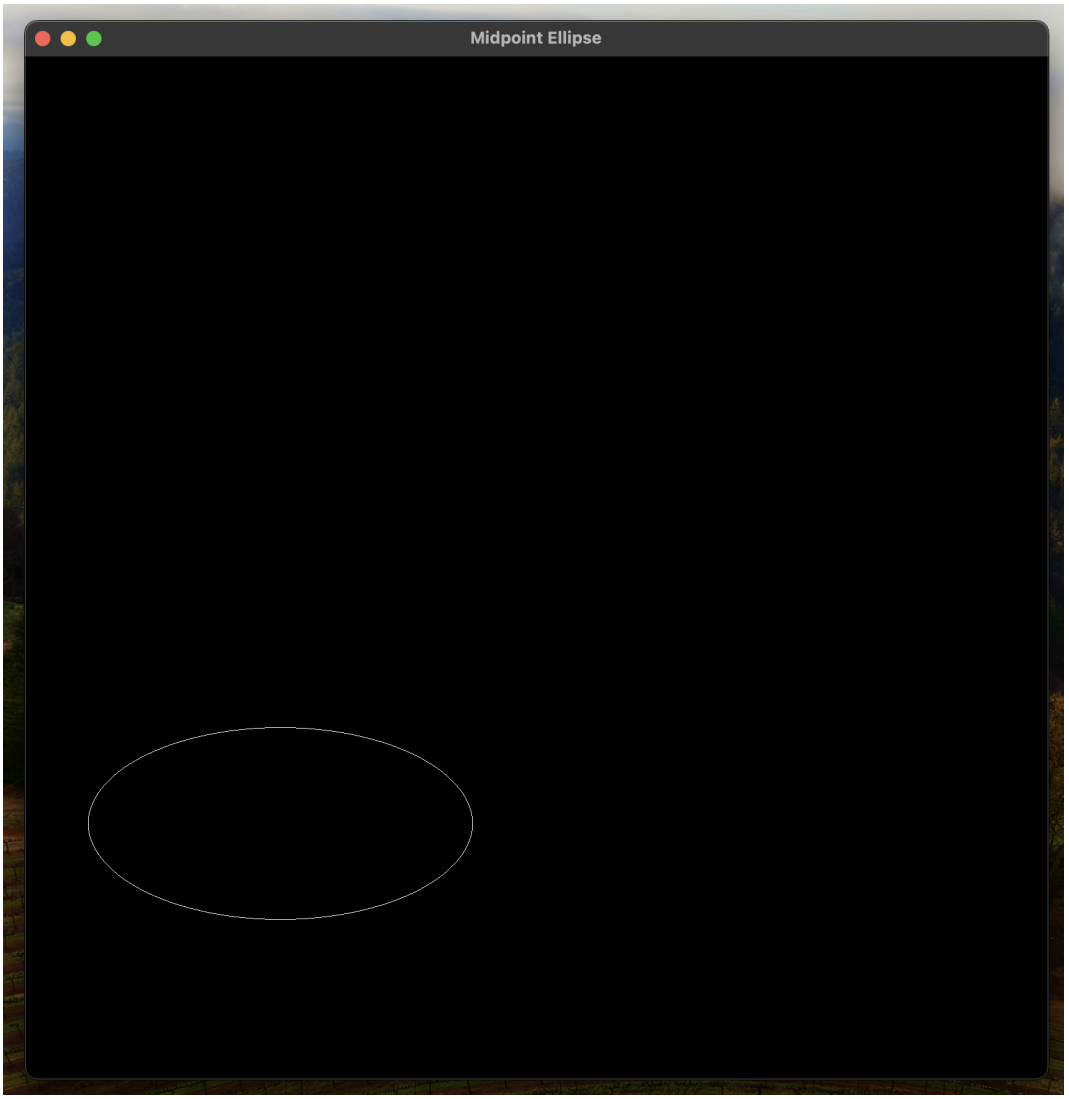
Midpoint Ellipse Algorithm Program:

The Mid-Point Ellipse Algorithm is a rasterization method utilized to compute the pixel coordinates necessary for drawing an ellipse. This algorithm accurately determines the perimeter points within the first quadrant and utilizes the ellipse's symmetry to mirror these points across the other three quadrants, ensuring a precise and complete depiction of the ellipse.

```python
import glfw
from OpenGL.GL import *
from OpenGL.GLU import *

win_width, win_height = 800, 800

def MidPointEllipse(x_center, y_center, rx, ry):
    points = []
    x = 0
    y = ry
    rx2 = rx * rx
    ry2 = ry * ry
    tworx2 = 2 * rx2
    twory2 = 2 * ry2

    # Initial decision parameter for region 1
    p1 = ry2 - (rx2 * ry) + (0.25 * rx2)

    # Region 1
    while tworx2 * y > twory2 * x:
        points.extend([(x_center + x, y_center + y), (x_center - x, y_center + y),
                       (x_center + x, y_center - y), (x_center - x, y_center - y)])
        if p1 < 0:
            x += 1
            p1 += twory2 * x + ry2
        else:
            x += 1
            y -= 1
            p1 += twory2 * x - tworx2 * y + ry2

    # Initial decision parameter for region 2
    p2 = ry2 * (x + 0.5)**2 + rx2 * (y - 1)**2 - rx2 * ry2

    # Region 2
    while y >= 0:
        points.extend([(x_center + x, y_center + y), (x_center - x, y_center + y),
                       (x_center + x, y_center - y), (x_center - x, y_center - y)])
        if p2 > 0:
            y -= 1
            p2 -= tworx2 * y + rx2
        else:
            x += 1
            y -= 1
            p2 += twory2 * x - tworx2 * y + rx2

    return points

```

```python
def draw_ellipse(x_center, y_center, rx, ry):
    points = MidPointEllipse(x_center, y_center, rx, ry)
    glBegin(GL_POINTS)
    glColor3f(1.0, 1.0, 1.0)  # Set color to white
    for x, y in points:
        glVertex2f(x, y)
    glEnd()

def main():
    # Initialize glfw
    if not glfw.init():
        return

    # Create a windowed mode window and its OpenGL context
    window = glfw.create_window(win_width, win_height, "Midpoint Ellipse", None,
None)if not window:
        glfw.terminate()
        return

    # Make the window's context current
    glfw.make_context_current(window)

    # Set viewport and projection
    glViewport(0, 0, win_width, win_height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(0, win_width, 0, win_height)

    # Loop until the user closes the window
    while not glfw.window_should_close(window):
        # Render here, e.g. using pyOpenGL
        glClear(GL_COLOR_BUFFER_BIT)
        draw_ellipse(400, 400, 300, 150)

        # Swap front and back buffers
        glfw.swap_buffers(window)

        # Poll for and process events
        glfw.poll_events()

    glfw.terminate()

if __name__ == "__main__":
    main()

```

Output:



This Python code uses GLFW and OpenGL to draw an ellipse with the Mid-Point Ellipse Algorithm. The MidPointEllipse function calculates the perimeter points of an ellipse centered at (x_center, y_center) with a major radius rx and a minor radius ry. The draw_ellipse function renders the ellipse using OpenGL.

In the main function, a GLFW window is initialized, and an ellipse is drawn at (400, 400) with a major radius of 300 and a minor radius of 150. The window refreshes until closed by the user.